**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

**ОТЧЁТ**

**по лабораторной работе №1**

**по дисциплине «Организация ЭВМ и систем»**

**Тема: ИССЛЕДОВАНИЕ ВНУТРЕННЕГО ПРЕДСТАВЛЕНИЯ РАЗЛИЧНЫХ ФОРМАТОВ ДАННЫХ**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3312 |  | Барченков П.А. |
| Студент гр. 3312 |  | Лебедев И.А. |
| Студент гр. 3312 |  | Шарапов И.Д. |
| Преподаватель |  | Ильин С.Е. |

Санкт-Петербург

2024

**Содержание**

[Описание задания 3](#_Toc178630694)

[Схема алгоритма 4](#_Toc178630695)

[Текст программы 8](#_Toc178630696)

[Примеры запуска программы 10](#_Toc178630697)

[Приложение 12](#_Toc178630698)

# Описание задания

Вариант 9.

1. Разработать алгоритм ввода с клавиатуры типов данных (*short int* и *float*) и показать на экране их внутреннее представление в двоичной системе счисления.
2. Написать и отладить программу на языке С++, реализующую разработанный алгоритм. Программа должна:

* иметь дружественный интерфейс
* выводить на экран информативное сообщение при вводе некорректных данных
* предложить повторный ввод пока не будут введены корректные данные.

1. Дополнить разработанный ранее алгоритм возможностью инвертировать значения всех бит кроме тех, количество и номера которых задаются с клавиатуры и последующего вывода преобразованного кода в двоичной системе счисления и в формате исходного данного.

# Схема алгоритма

Блок кода, представленный на рисунке 1, демонстрирует работу функции main, в которой мы считываем, обрабатываем и выводим числа. Сначала проводим полную обработку ShortInt, затем Float.

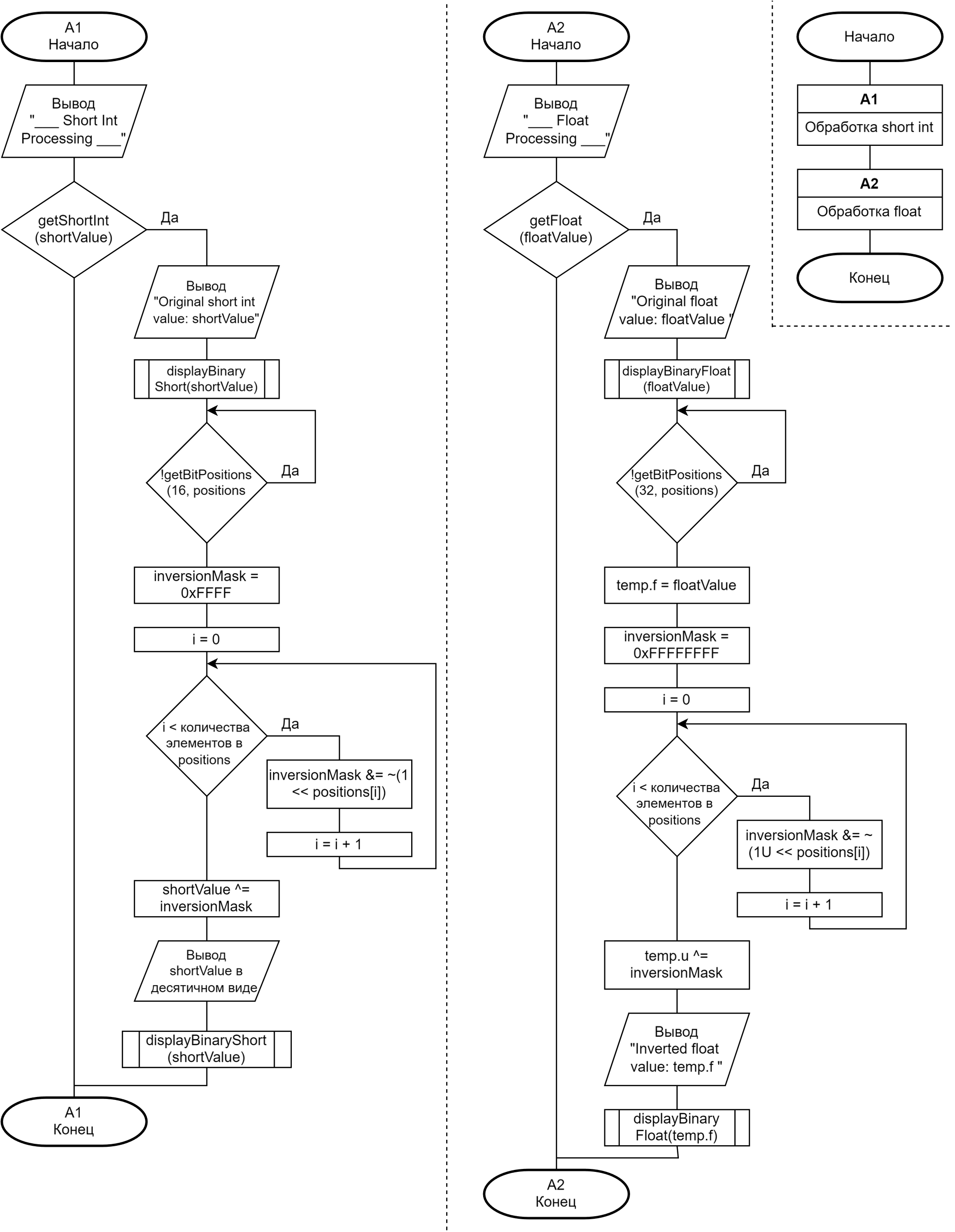


Рисунок 1 – Cхема функции main

Блок кода, представленный на рисунке 2, демонстрирует работу функции getBitPosition, в которой происходит обработка некорректного ввода пользователем и считывание индексов тех битов, которые не нужно инвертировать.

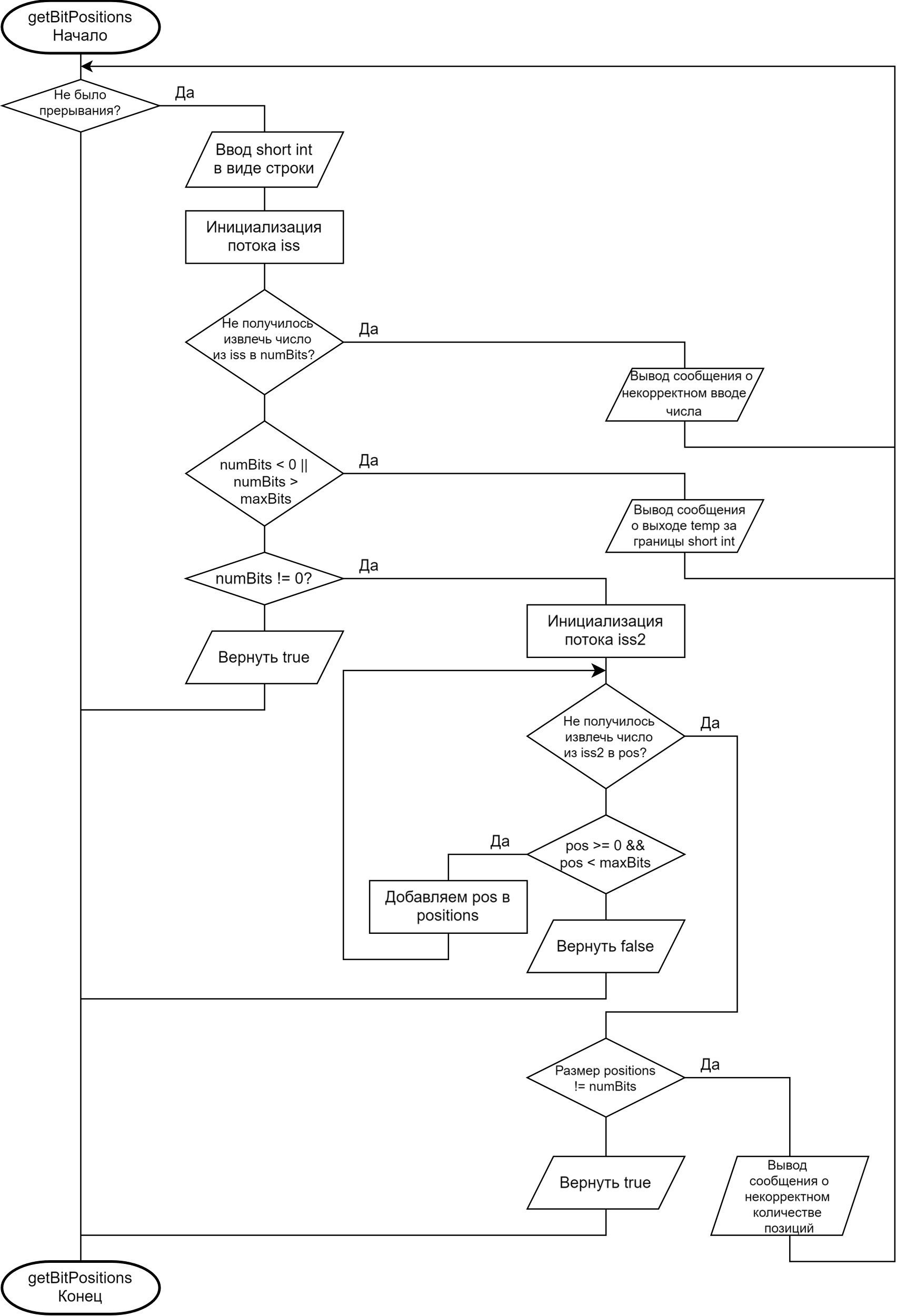


Рисунок 2 – Схема функции getBitPosition

Блок кода, представленный на рисунке 3, демонстрирует работу двух функций для обработки ShortInt. getShortInt используется для обработки некорректного ввода пользователем и считыванием самого числа. displayBinaryShort для вывода на экран двоичного представления полученного числа.

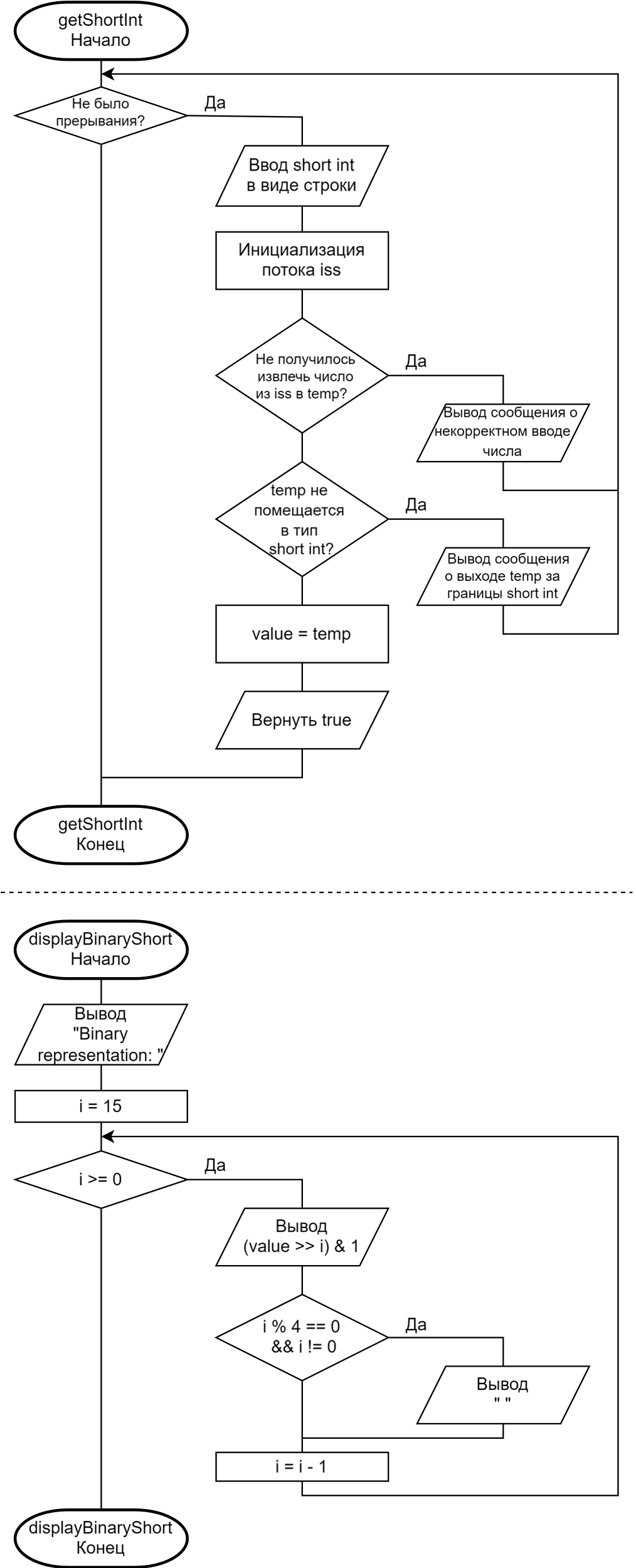


Рисунок 3 – Схема функций для работы с ShortInt

Блок кода, представленный на рисунке 4, демонстрирует работу двух функций для обработки Float. getFloat используется для обработки некорректного ввода пользователем и считыванием самого числа. displayBinaryFloat для вывода на экран двоичного представления полученного числа.

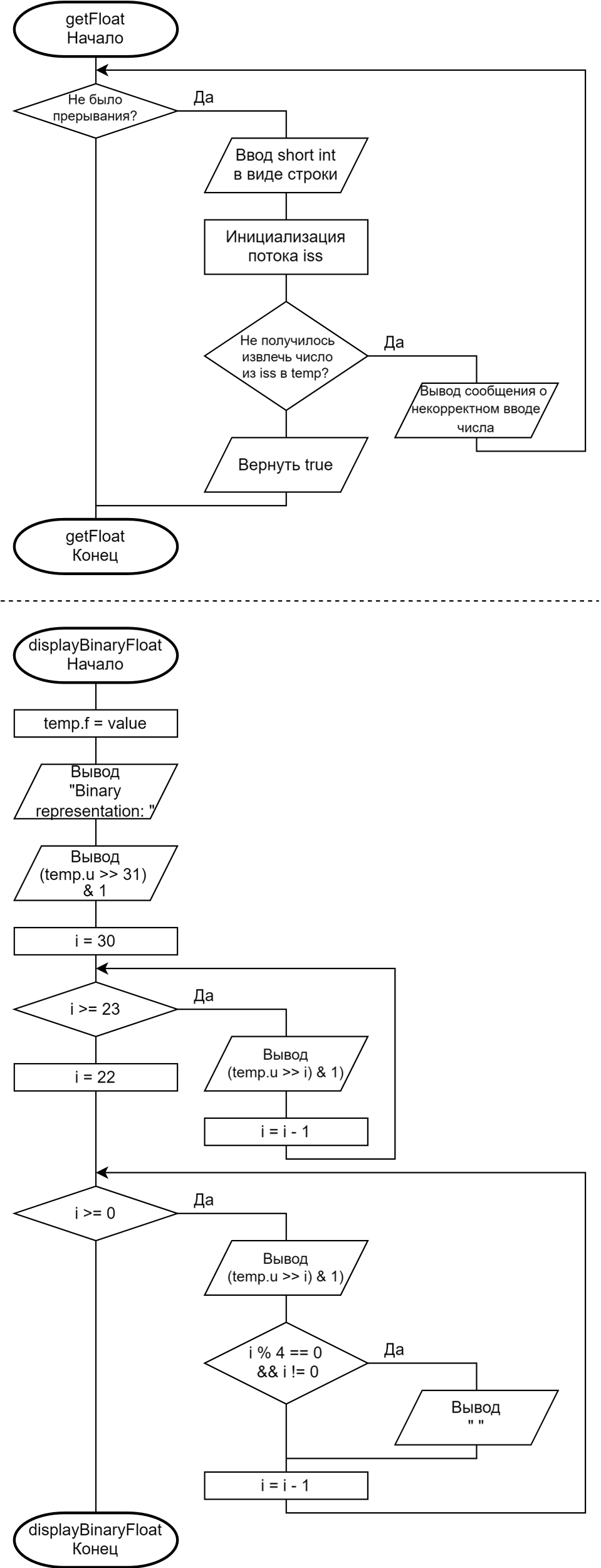


Рисунок 4 - Схема функций для работы с Float

# Текст программы

|  |
| --- |
| #include <iostream> #include <limits> #include <string> #include <sstream> #include <vector>  void displayBinaryShort(short int value) {  std::cout << "Binary representation: ";  for (int i = 15; i >= 0; --i) {  std::cout << ((value >> i) & 1);  if (i % 4 == 0 && i != 0) std::cout << " ";  }  std::cout << std::endl; }  void displayBinaryFloat(float value) {  union {  float f;  unsigned int u;  } temp;  temp.f = value;   std::cout << "Binary representation: ";   // Print the sign bit (1 bit)  std::cout << ((temp.u >> 31) & 1) << " ";   // Print the exponent bits (8 bits)  std::cout << "(";  for (int i = 30; i >= 23; --i) {  std::cout << ((temp.u >> i) & 1);  }  std::cout << ") ";   // Print the mantissa bits (23 bits)  std::cout << "(";  for (int i = 22; i >= 0; --i) {  std::cout << ((temp.u >> i) & 1);  if (i % 4 == 0 && i != 0) {  std::cout << " ";  }  }  std::cout << ")" << std::endl; }  bool getShortInt(short int & value) {  while (true) {  std::cout << "Please enter a short int value (from -32768 to 32767):";  std::string input;  std::getline(std::cin, input);   std::istringstream iss(input);  int temp;  if (iss >> temp) {  if (temp >= std::numeric\_limits<short int>::min() &&  temp <= std::numeric\_limits<short int>::max()) {  value = static\_cast<short int>(temp);  return true;  } else {  std::cout << "Value out of range for short int. Please try again (from -32768 to 32767)." << std::endl;  }  } else {  std::cout << "Invalid input. Please enter a valid short int number." << std::endl;  }  } }  bool getFloat(float & value) {  while (true) {  std::cout << "Please enter a float value:";  std::string input;  std::getline(std::cin, input);   std::istringstream iss(input);  if (iss >> value) {  return true;  } else {  std::cout << "Invalid input. Please enter a valid float number." << std::endl;  }  } }  bool getBitPositions(int maxBits, std::vector<int>& positions) {  while (true) {  positions.clear();  std::cout << "Please enter the number of bits to exclude from inversion (0 to " << maxBits << "):";  std::string input;  std::getline(std::cin, input);  std::istringstream iss(input);  int numBits;  if (iss >> numBits) {  if (numBits >= 0 && numBits <= maxBits) {  if (numBits == 0) return true;  std::cout << "Enter the bit positions to exclude (separated by spaces, between 0 and " << (maxBits - 1) << "):";  std::getline(std::cin, input);  std::istringstream iss2(input);  int pos;  while (iss2 >> pos) {  if (pos < 0 || pos >= maxBits) {  std::cout << "Bit position " << pos << " is out of range. Please enter positions between 0 and " << (maxBits - 1) << "." << std::endl;  return false;  }  positions.push\_back(pos);  }  if (positions.size() != static\_cast<size\_t>(numBits)) {  std::cout << "Number of positions entered does not match the number specified. Please try again." << std::endl;  continue;  }  return true;  } else {  std::cout << "Invalid number of bits to exclude. Please enter a number between 0 and " << maxBits << "." << std::endl;  }  } else {  std::cout << "Invalid input. Please enter a valid number." << std::endl;  }  } }  int main() {  std::cout << "\_\_\_ Short Int Processing \_\_\_" << std::endl;  short int shortValue;  if (getShortInt(shortValue)) {  std::cout << "Original short int value: " << shortValue << std::endl;  displayBinaryShort(shortValue);   std::vector<int> positions;  while (!getBitPositions(16, positions)) {  }   unsigned short inversionMask = 0xFFFF;  for (size\_t i = 0; i < positions.size(); ++i) {  inversionMask &= ~(1 << positions[i]);  }   shortValue ^= inversionMask;  std::cout << "Inverted short int value: " << shortValue << std::endl;  displayBinaryShort(shortValue);  }   std::cout << "\n\_\_\_ Float Processing \_\_\_" << std::endl;  float floatValue;  if (getFloat(floatValue)) {  std::cout << "Original float value: " << floatValue << std::endl;  displayBinaryFloat(floatValue);   std::vector<int> positions;  while (!getBitPositions(32, positions)) {  }   union {  float f;  unsigned int u;  } temp;  temp.f = floatValue;   unsigned int inversionMask = 0xFFFFFFFF;  for (size\_t i = 0; i < positions.size(); ++i) {  inversionMask &= ~(1U << positions[i]);  }   temp.u ^= inversionMask;  std::cout << "Inverted float value: " << temp.f << std::endl;  displayBinaryFloat(temp.f);  }   return 0; } |

# Примеры запуска программы

В первом примере (Рис. 5) демонстрируется обработка ошибочного ввода исходого числа пользователем.

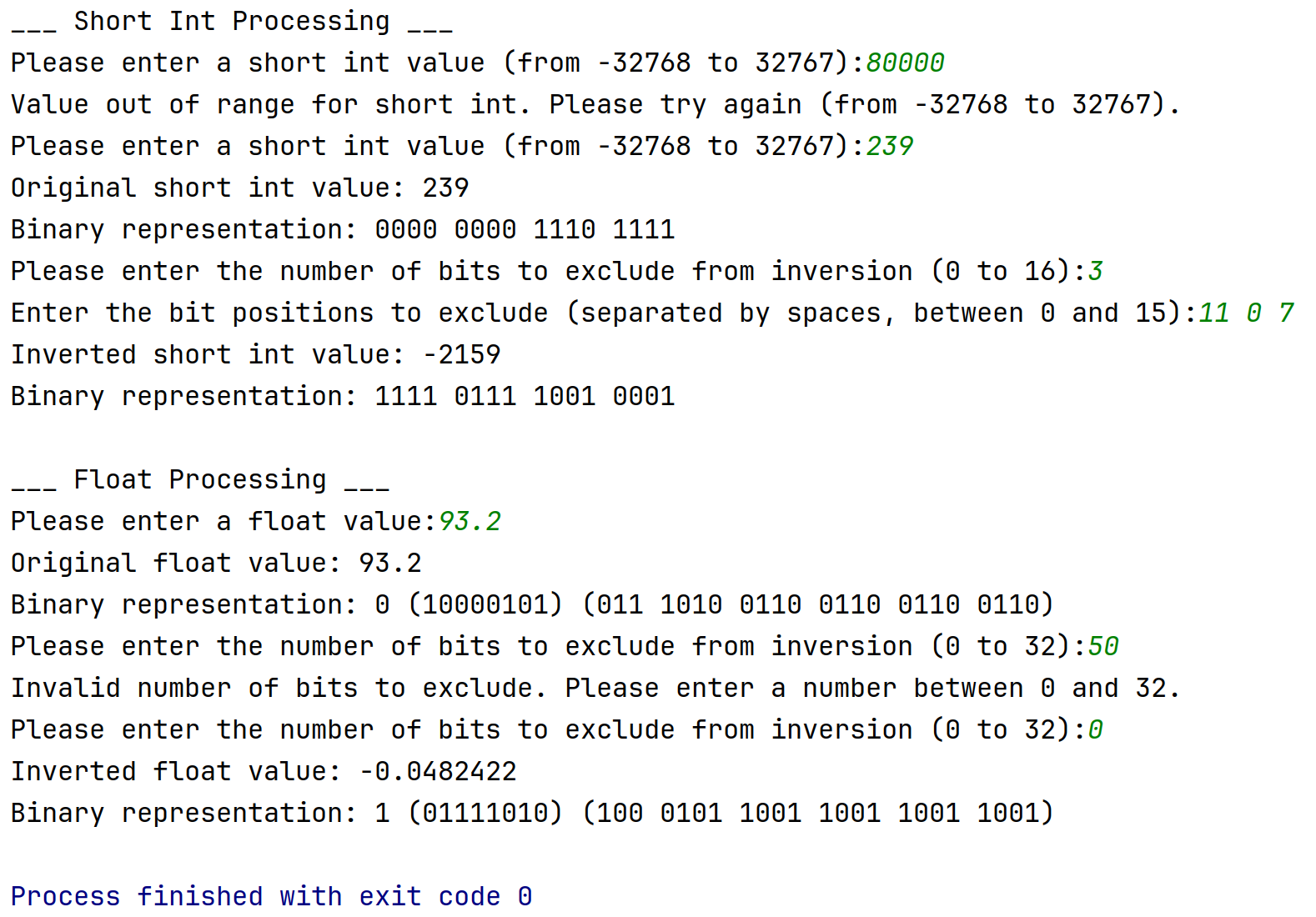


Рисунок 5 – Пример с ошибочным вводом в первом поле

Во втором примере (Рис. 6) демонстрируется обработка большого числа, неправильного ввода индексов пользователем и вводом букв там, где требуется число.

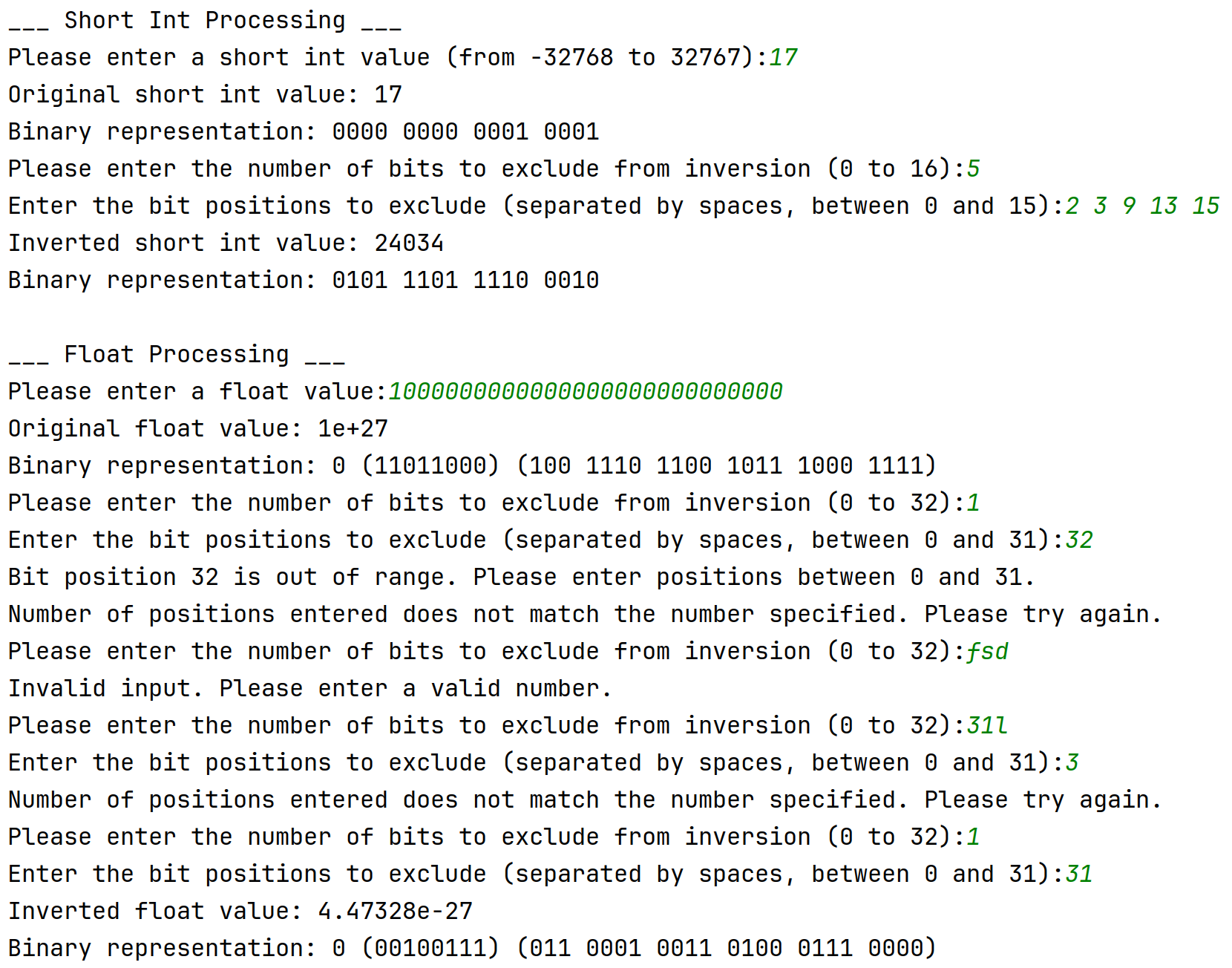


Рисунок 6 – Пример с ошибным вводом во втором и третьем поле

В третьем примере (Рис. 7) демонстрируется работа программы с полностью корректными данными, введёнными пользователем.

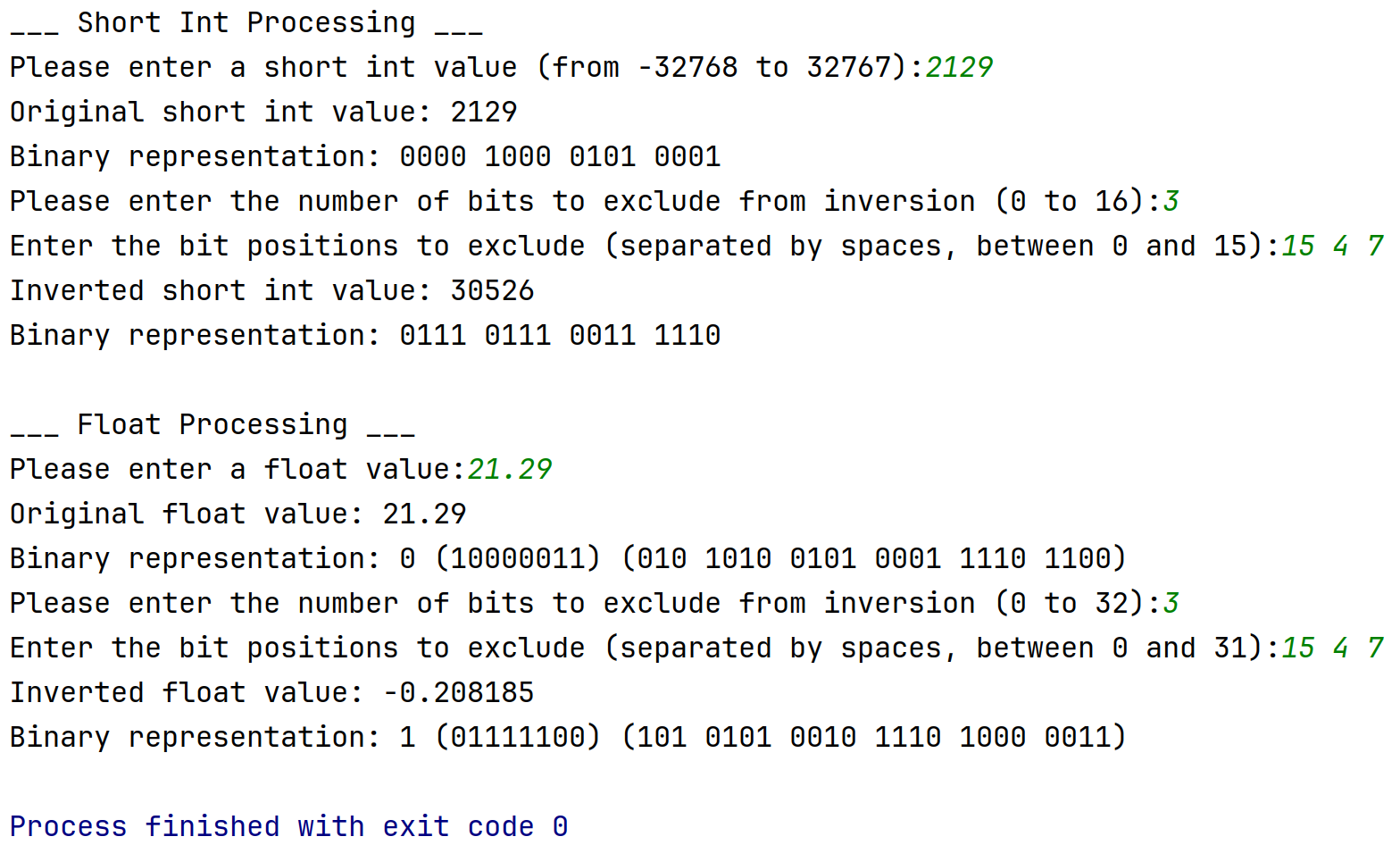


Рисунок 7 – Пример без ошибок

# Приложение

В примерах программа выполнялась в среде разработки JetBrains CLion 2024.2.1., стандартом языка С++ 98 и под управлением ОС Linux Mint 22.